# Robotics Practicals

# Topic 4: Industrial SCARA Robot Adept

Jan FROGG, Sascha FREY, Stanislas FURRER

May 20, 2019

## 1 Introduction

This report will summarize some of the assimilated concepts during the practical and present the strategies used to complete the assigned task. This task consists primarily of understanding the required programming to make the robot execute well-defined movements as well as creating a small program which can be used to play tic-tac-toe by two players.

## 2 Theoretical aspects of the SCARA Adept robot

We will summarize three key concepts that were learnt during the practical. These are; the primary differences between a serial robot (such as the Adept Cobra) and a parallel robot, the relationship between joint positions and the end-effector position, as well as the different solutions to guarantee operator safety around an industrial robot that is typically able to cause significant damage.

### 2.1 Serial vs. Parallel Robots

The differences between serial and parallel robots can generally be summarized by three points.

- Stiffness
- Error Propagation
- Inertia

The stiffness of parallel robots is generally higher. Each parallel branch adds to the overall stiffness of the robot rather than reducing it as is the case of an additional segment in a serial robot. This is clearly visible in the construction of parallel and serial robots. For example, the Adept Cobra has segments that are very thick, and therefore heavy, in order to provide the desired overall rigidity. As a counter example, the Delta Robot, a parallel robot, has very thin segments connecting the end-effector to the base, and has a similar overall rigidity for an almost identical application: picking and placing objects of small mass.

The error propagation follows a similar trend to the above-mentioned stiffness. For a serial robot, the error in each of the joints adds up to give a larger error at the end-effector position. This means

that each of the actuators needs to be more precise for a given desired tolerance. In a parallel robot, the opposite is true. The errors of the joints at the base cancel each other out to some degree, or at the very worst case don't add up, they are rather averaged. This means that, for a given motor precision a parallel robot is able to complete tasks more precisely, which is why they are applied extensively in high-precision industries such as watchmaking or microelectronics.

Finally, the inertia of serial robots, such as a scara robot is much larger than that of a parallel robot. This is, to a minor extent, due to the larger segments that are required to obtain a given rigidity. The main reason, however, is the weight of the motors at each of the joint positions. The motor on the base joint has to rotate the inertia of all subsequent motors up to the end-effector. This high inertia implies that motors have to have high torques to get the desired acceleration. A parallel robot is typically able to have all motors attached to the base. A good example is the delta robot which has three motors attached to the base and at most a single motor for the rotation of the end-effector.

## 2.2 Joint angle and end-effector position

As is visible in the ACE programming environment there is a distinction between the robot joint angles ($J_1$,$J_2$,$J_3$,$J_4$) and the X,Y,Z,RZ positions/angle of the end-effector. We note here the direct kinematic model which gives the end-effector position in terms of the joint angles and the inverse kinematic model, which as the name suggests is the opposite.

We use the conventions visible

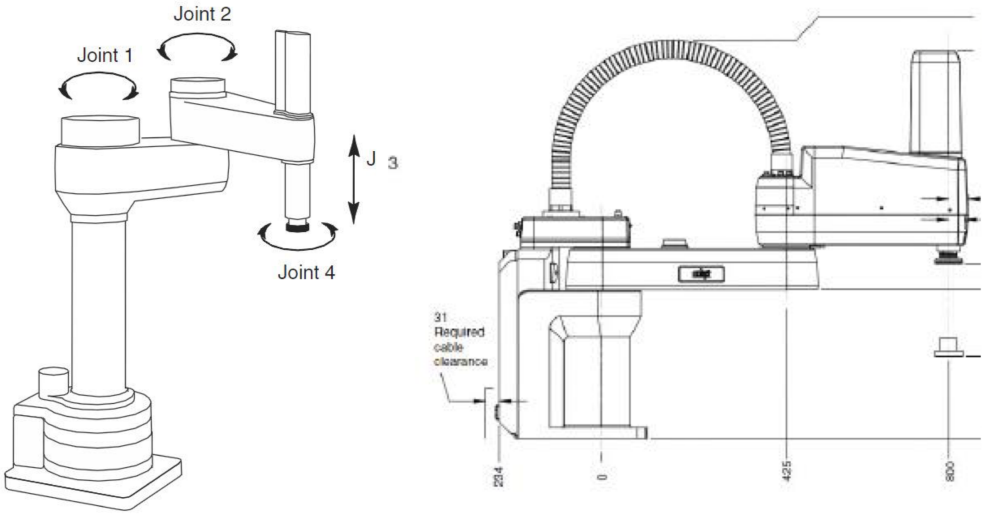


Figure 1: Adept Cobra robot sketch, showing the lengths and the joints of the joints. The right-hand-side image shows the robot position when all joint are at 0.

### 2.2.1 Direct Kinematic Model

$$x = L_1 \times cos(J_1) + L_2 \times cos(J_1 + J_2) \quad (1)$$
$$y = L_1 \times sin(J_1) + L_2 \times sin(J_1 + J_2) \quad (2)$$
$$z = J_3 + z* \quad (3)$$
$$rz = J_1 + J_2 + J_4 \quad (4)$$

Where:

$$L_1 = 425mm \quad (5)$$
$$L_2 = 800 - 425 = 375mm \quad (6)$$
$z*$ is a given offset in the z axis (7)

### 2.2.2 Inverse Kinematic Model

Using [1] as an inspiration for the inverse kinematic calculation, we can note the joint angles in terms of the X,Y,Z,RZ coordinates. Note that the nature of the problem means that there are multiple solutions to this problem. In the case of a SCARA robot this may be in bending the "elbow" in two different ways as indicated in Figure 2.

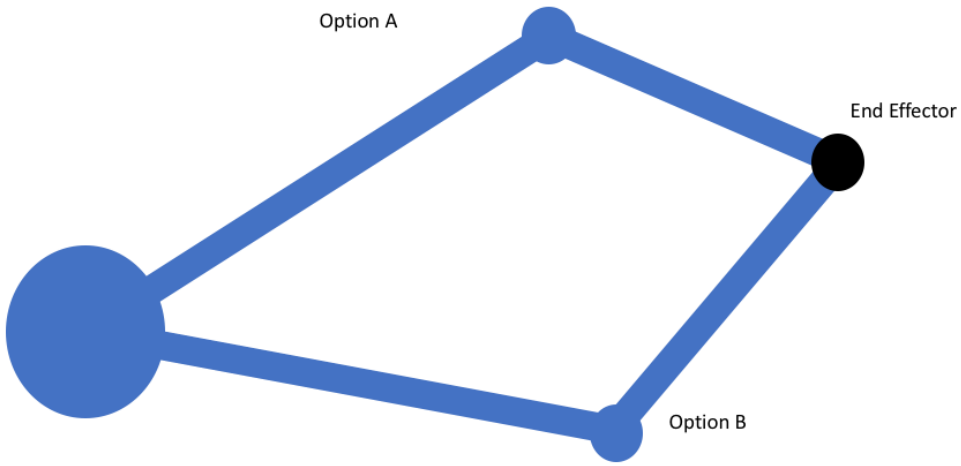


Figure 2: Example for the SCARA where two different solutions of the inverse kinematic problem. Only one solution is given in the report.

$$J_1 = atan(\frac{y}{x}) - cos^{-1}(\frac{x^2 + y^2 + L_1^2 - L_2^2}{2L_1\sqrt{x^2 + y^2}}) \tag{8}$$

$$J_2 = cos^{-1}\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2}) \tag{9}$$

$$J_3 = z - z* \tag{10}$$

$$J_4 = rz - J_1 - J_2 \tag{11}$$

### 2.3 Safety Concerns and Preventative Measures

The requirement for high speeds and accelerations on a serial robot with high inertia implies that significant torque is created by the motors. This can strongly damage operators that may come into contact with the arms during operation. There are a number of ways to ensure operator safety:

- Caging the Motor entirely
- Using proximity sensors
- Intense operator training
- Contact sensors / Collaborative robots

These solutions vary in level of protection but also in the restriction of motor uses.
Caging the motor entirely is the 100% safe solution as the operator physically cannot come into contact with the robot. This brings with it the disadvantage that the operator cannot clearly see what is going on which will reduce the quality control possibilities.
Using proximity sensors means that the operator can have a clear line of sight to the work space but is sufficiently safe in the sense that he or she cannot enter the work space without the robot stopping immediately.
Relying solely on operator training is the final solution which allows for greatest flexibility but provides minimal safety as the operator must personally ensure he or she does not enter the work space.
The final solution is an emerging solution in robotics, using a collaborative approach were the operator and the robot can coexist in the same work space. This can be done slowing down the robot significantly which is often opposite to the purpose of the robot. The other solution is a state-of-the-art motor which is able to interpret contact very quickly and react before causing damage, proposed by Genesis Robotics [2] as seen in the course "Industrial and Applied Robotics" by Prof. Bouri.

## 3 Tic-tac-toe game

In this section we show in details how the tic-tac-toe game was implemented. The game is played as follows: The program shows a prompt window for each player to input the position he/she wishes to place his/her metal puck on. The robot then picks the puck from the bench to place it at the specified position. Therefore, we had to implement the method for this pick and place motion, as well as write the code specifying the target positions for this motion.

Firstly, for the robot to pick and place the pieces in an intelligent manner, it must know their positions on the bench, as well as on the board. Therefore, we first specified the position of each piece to be picked one by one in a logical order (from first to last, vertically) for each player's side.

Then the coordinates of each possible position on the board was recorded for the robot to know where to place the pucks. It is important to note that all of this was done by recording only one main position and then, thanks to the known spacing between each position on the board, we were able to just add this spacing to the original position. This avoids the time consuming method of measuring each individual position and then record all of them.

The pick and place motion was implemented based on one main constraint, which is the fact that the robot must be able to perform the entire motion without running into any obstacle. That is, it must be able to pick the pucks without hitting the others and especially it should not disturb the pucks' distribution on the game board. In order to do this, we made sure that the end effector of the robot approached a small neighbourhood of each piece at an appropriate height (far enough to not touch any of the pieces) before triggering the end effector's descent towards them. Figures 3 shows the chosen strategy for the tic-tac-toe game.

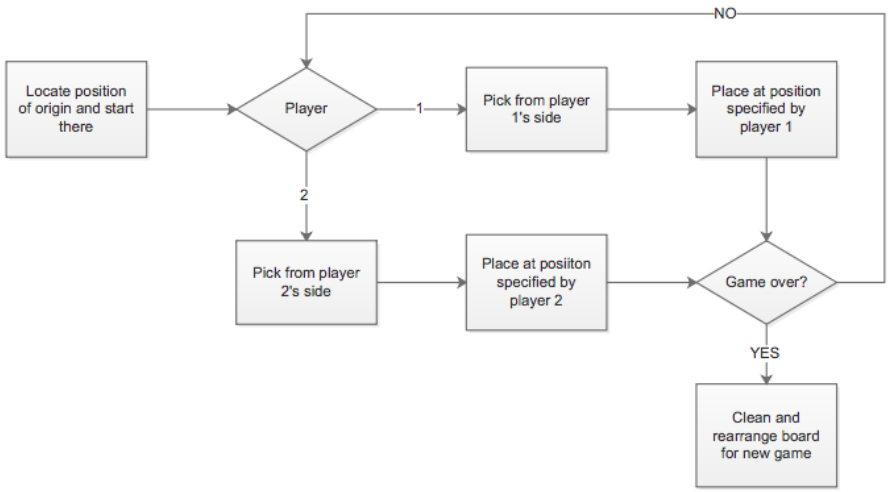


Figure 3: Basic flowchart describing the strategy used for the game to be played.

# 4 Go, pick and place algorithm

The game frame being set, we were able to localize every pawn on each side and knew to which position the robot should displace them during the game. In this section, we will explain how we implemented an algorithm that allows the robot to go to the selected pawn, pick it and place it to the right place. Then the robot arm repositions itself at the origin of the frame to be able to restart the pick and place algorithm until the end of the game.

The commands APPRO and MOVE allow us respectively to move freely on the game frame without smashing any pawn on the way and go directly to the desired position. The command BREAK was used after each action to get smooth trajectory. The robot would complete its trajectory entirely before going to the next line. In our setup case, the command SIGNAL and the timer was not necessary since we didn't have pneumatic actuator. Figure 4 shows the chosen strategy for the pick and place motions.

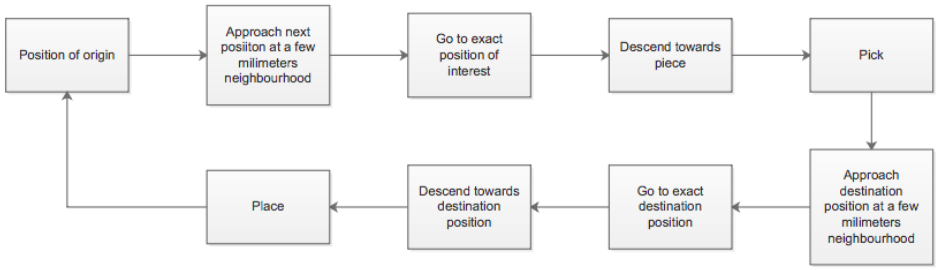


Figure 4: Basic flowchart describing the strategy used for the pick and place motions.

# 5 Conclusion

This practical was pretty good because in 4 hours it is possible to get familiar with the robot, ACE software and the way to code the algorithm. It shows how intuitive industrial robotics can be. Unfortunately we did not have the time to go in real depth into the bonus question. A basic algorithm with which the robot could rearrange the entire board would be to perform the exact inverse operations, which means each piece would be replaced at its original position (on each player's side) from the last on the first one that was placed in the beginning of the game. Although it is clear there must be a more intelligent solution to cleaning the board before starting a new game, this one is still a solution that would work.

The only change that would be great, would be to add an extra 4 hour for doing the same with crepes.

# References

[1] Application Note (2004) *SCARA Robot Kinematics*, Available from: http://www.deltatau.com/Common/technotes/SCARA%20Robot%20Kinematics.pdf Accessed on 20 May 2019.

[2] Genesis Robotics : http://www.genesis-robotics.com